TRƯỜNG ĐẠI HỌC ĐẠI NAM

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

TÊN MÔN HỌC: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

TÊN ĐỀ TÀI: CÀI ĐẶT DANH SÁCH LIÊN KẾT ĐƠN TRÊN MẢNG <1>

Giảng viên hướng dẫn: TS. Trần Quý Nam

Sinh viên thực hiện: Lê Quốc Duy

Nguyễn Xong Phi

Đặng Việt Hoàng

Hà Nội, 2025

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

TÊN MÔN HỌC: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

TÊN ĐỀ TÀI: CÀI ĐẶT DANH SÁCH LIÊN KẾT ĐƠN TRÊN MẢNG <1>

STT	Mã Sinh Viên	Họ và Tên	Ngày Sinh	Điểm	
				Bằng Số	Bằng Chữ
1	1871040012	Lê Quốc Duy	05/10/2006		
2	1871040025	Nguyễn Xong Phi	18/09/2005		
3	1871040015	Đặng Việt Hoàng	19/10/2005		

CÁN BỘ CHẤM THI 1 **CÁN BỘ CHẤM THI 2**

Hà Nội, 2025

LỜI NÓI ĐẦU

Để hoàn thành được bài tiểu luận này, em xin chân thành cảm ơn Ban Giám hiệu, các khoa, phòng và quý thầy, cô của trường đại học Đại Nam, những người đã tận tình giúp đỡ và tạo điều kiện cho em trong quá trình học tập. Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến thầy **TS. Trần Quý Nam**- người đã trực tiếp giảng dạy và hướng dẫn em thực hiện bài tiểu luận này bằng tất cả lòng nhiệt tình và sự quan tâm sâu sắc.

Trong quá trình thực hiện bài tiểu luận này, do hiểu biết còn nhiều hạn chế nên bài làm khó tránh khỏi những thiếu sót. Em rất mong nhận được những lời góp ý của quý thầy cô để bài tiểu luận ngày càng hoàn thiện hơn.

Sinh viên thực hiện

Nguyễn Xong Phi

Lê Quốc Duy

Đặng Việt Hoàng

LÝ DO CHỌN ĐỀ TÀI

Danh sách liên kết đơn là một cấu trúc dữ liệu cơ bản và thường được triển khai bằng con trỏ. Tuy nhiên, việc cài đặt danh sách liên kết đơn trên mảng mang lại một cách tiếp cận mới mẻ, giúp hiểu rõ hơn về cách hoạt động của các cấu trúc dữ liệu thông qua việc tận dụng chỉ số để mô phỏng liên kết.

Chủ đề này không chỉ củng cố kiến thức về cấu trúc dữ liệu mà còn phát triển tư duy sáng tạo trong việc tổ chức và quản lý dữ liệu. Hơn nữa, nó phù hợp với các tình huống khi việc sử dụng bộ nhớ động (dynamic memory) không khả thi, từ đó mở rộng khả năng ứng dụng trong thực tế.

Nhóm chúng em chọn chủ đề này với mong muốn cung cấp một góc nhìn mới mẻ và sáng tạo, đồng thời tạo cơ hội để các thành viên nâng cao hiểu biết về cấu trúc dữ liệu, áp dụng hiệu quả trong học tập và thực tiễn.

MỤC LỤC

MỤC LỤC HÌNH ẢNH

BẢNG CÁC TỪ VIẾT TẮT

STT	TỪ VIẾT TẮT	VIẾT ĐẦY ĐỦ
1	DSLK	Danh Sách Liên Kết

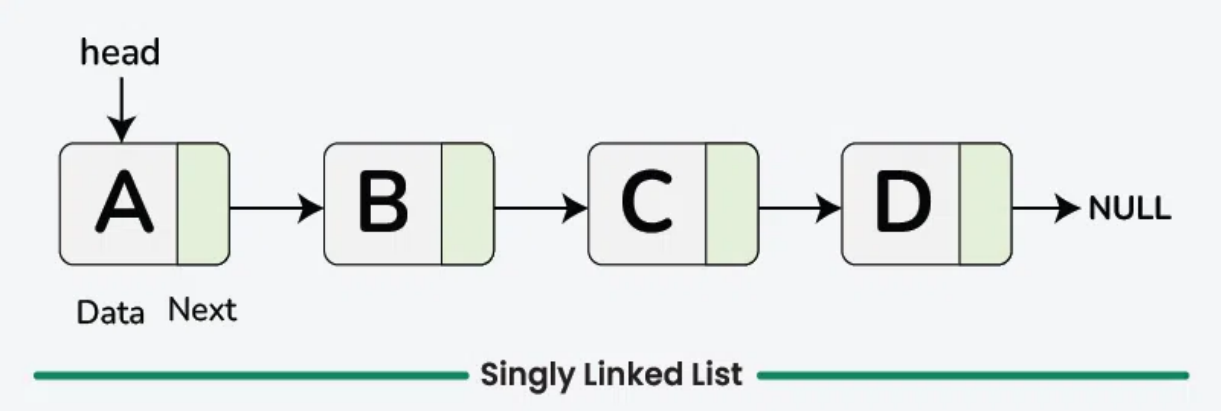
CHƯƠNG I: DANH SÁCH LIÊN KẾT ĐƠN

1.1. DANH SÁCH LIÊN KẾT ĐƠN LÀ GÌ ?

Danh sách liên kết đơn là một cấu trúc dữ liệu cơ bản, nó bao gồm các **nodes**(nút) trong đó mỗi nút chứa trong đó mỗi nút chứa một **trường dữ liệu** và tham thiếu tới nút tiếp theo của danh sách liên kết. Nút tiếp theo của nút cuối cùng là **null,** cho biết sự kết thúc của danh sách. Danh sách liên kết hỗ trợ các thao tác **chèn** và **xóa** hiệu quả.

1.2. CẤU TRÚC CỦA MỘT NODE.

Trong một danh sách liên kết đơn, mỗi node bao gồm hai phần : dữ liệu và một con trỏ tới nút tiếp theo. Cấu trúc này cho phép các node được liên kết động với nhau tạo thành một chuỗi giống như một dãy liên tiếp.



Hình 1.1: Danh sách liên kết đơn

```
// Định nghĩa một nút trong danh sách liên kết đơn
struct Node {
// Phần dữ liệu của nút
    int data;
// Con trỏ trỏ đến nút tiếp theo trong danh sách
    Node* next;
// Constructor để khởi tạo nút với dữ liệu
    Node(int data)
    {
        this->data = data;
        this->next = nullptr;
    }
};
```

Trong ví dụ này lớp node chứa một trường dữ liệu số nguyên lưu trữ thông tin và một con trỏ tới node khác tạo liên kết tới nút tiếp theo trong danh sách.

1.3. CÁC THAO TÁC TRÊN DANH SÁCH LIÊN KẾT ĐƠN

- **Duyệt**
- **Tìm kiếm**
- **Tìm độ dài**
- **Chèn : chèn vào đầu, chèn vào cuối, chèn vào một vị trí cụ thể**
- **Xóa: xóa phần tử đầu, xóa phần tử cuối, xóa một nút cụ thể**

Được rồi hãy đi qua từng thao tác được nhắc đến ở trên, từng cái một.

CHƯƠNG II: DUYỆT DANH SÁCH LIÊN KẾT ĐƠN

2.1. DUYỆT DANH SÁCH

- Duyệt bao gồm việc truy cập từng nút trong DSLK đơn và thực hiện một số thao tác trên dữ liệu.
- Một hàm duyệt đơn giản sẽ in hoặc xử lý dữ liệu của từng nút

Các bước tiếp cận:

- Khởi tạo một con trỏ hiện tại ở đầu danh sách
- Sử dụng một vòng lặp while lặp qua danh sách cho đến khi con trỏ hiện tại đạt giá trị NULL
- Bên trong vòng lặp, in dữ liệu của nút hiện tại và di duyển con trỏ hiện tại sang nút tiếp theo

2.2. HÀM DUYỆT DSLK ĐƠN

```
// Hàm C++ duyệt và in phần tử của danh sách liên kết

    void traverseLinkedList (Node* head){

// Bắt đầu từ đầu danh sách liên kết

        Node* current = head;

// Duyệt danh sách liên kết cho đến khi kết thúc

// (nullptr)

        while (current != nullptr) {

// In dữ liệu của node hiện tại

        cout << current->data << " ";

// Di chuyển đến node tiếp theo

        current = current->next;

    }

    cout << std::endl;

    }
```

CHƯƠNG III: TÌM KIẾM TRONG DANH SÁCH LIÊN KẾT ĐƠN

3.1. TÌM KIẾM TRONG DSLK ĐƠN NHƯ THẾ NÀO?

Tìm kiếm trong một danh sách liên kết đơn đề cập tới quá trình tìm kiếm một phần tử cụ thể hoặc giá trị bên trong các phần tử của danh sách liên kiết.

Các bước tiếp cận

1. Duyệt DSLK bắt đầu từ phần tử đầu
2. Kiểm tra dữ liệu của nút hiện tại có khớp với giá trị cần tìm hay không.

- Nếu có return **true**.

3. Nếu không thì, di chuyển tới node tiếp theo và lặp lại bước 2
4. Nếu tới cuối của danh sách mà không có kết quả nào phù hợp thì return **false**

3.2. HÀM TÌM KIẾM TRONG DSLK ĐƠN

```
// Hàm tìm kiếm một giá trị trong danh sách liên kết

bool searchLinkedList(struct Node* head, int target)

// Duyệt danh sách liên kết

        while (head != nullptr) {

// Kiểm tra nếu nút hiện tại

// trùng khớp với giá trị cần tìm

                if (head->data == target) {

                        return true; // Tìm thấy return true

                }

// Di chuyển đến nút tiếp theo

                head = head->next;

        }

        return false; // Không tìm thấy return false

}
```

CHƯƠNG IV: ĐỘ DÀI CỦA DSLK ĐƠN

4.1. TÌM ĐỘ DÀI DSLK ĐƠN NHƯ THẾ NÀO?

Tìm độ dài trong danh sác liên kết đơn đề cập tới quá trình xác định tổng số node trong một DSLK đơn.

Các bước tiếp cận:

- Khởi tạo một biến đếm **length** từ 0.
- Bắt đầu từ đầu của danh sách, gán nó cho hiện tại
- Duyệt danh sách

  + Tăng **length** sau mỗi node.

  + Di chuyển tới node tiếp theo **(current = current->)**

- Return kết quả cuối cùng của **length**

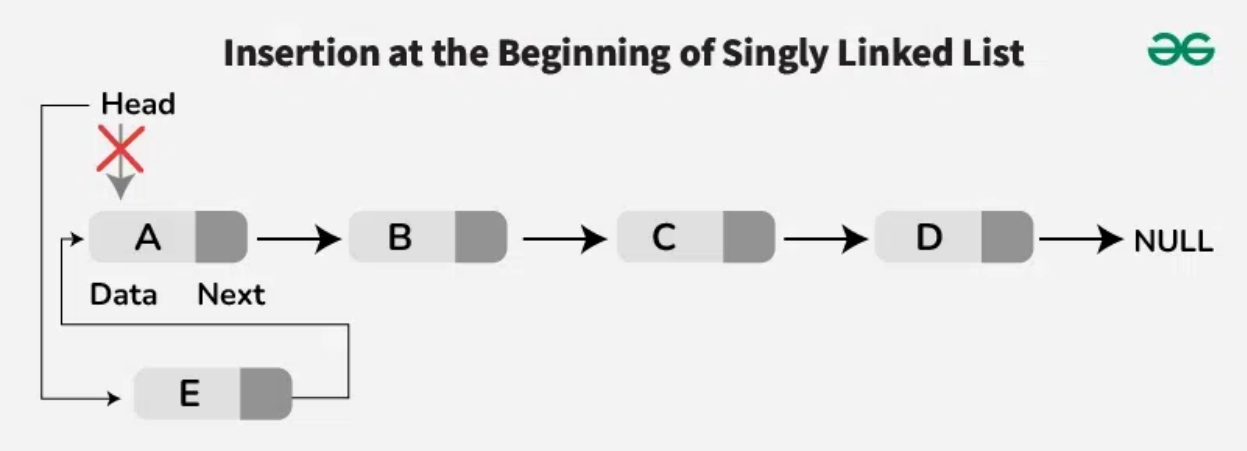
4.2. HÀM TÌM ĐỘ DÀI TRONG DSLK ĐƠN

```
// Hàm C++ tìm độ dài của danh sách liên kết

int findLength (Node* head){

// Initialize a counter for the length

// Khởi tạo biến đếm length

    int length = 0;

// Bắt đầu từ đầu danh sách

    Node* current = head;

// Duyệt danh sách tăng length sau mỗi

 // node

    while (current != nullptr) {

        length++;

        current = current->next;

    }

// Return độ dài cuối cùng của danh sách liên kết

    return length; }
```

Chương V: CHÈN TRONG DSLK ĐƠN

Chèn là một thao tác cơ bản trong DSLK liên quan đến việc thêm một nút mới vào danh sách. Có một số tình huống để chèn:

5.1.CHÈN VÀO ĐẦU DSLK ĐƠN



Hình 1.2: *Chèn vào đầu DSLK đơn*

Các bước tiếp cận:

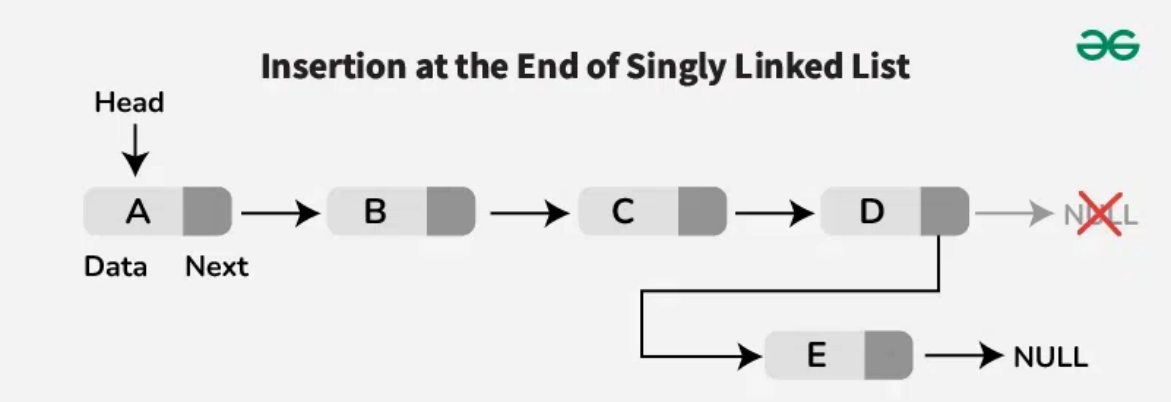
- Tạo một node mới với giá trị đã cho
- Đặt con trỏ **next** của nút mới trỏ đến nút hiện tại(head).
- Di chuyển con trỏ **head** để trỏ đến nút mới
- Return nút đầu (head) mới của DSLK

5.2. BÊN DƯỚI LÀ HÀM CHÈN ĐẦU DSLK

```
// Hàm C++ chèn một nút mới vào đầu

// DSLK

Node* insertAtBeginning (Node* head, int value)

{

// Tạo một nút mới vớ giá trị đã cho

        Node* newNode = new Node(value);

// Đặt con trỏ next của nút mới vào nút hiện tại

 // head

        newNode->next = head;

// Di chuyển con trỏ head trỏ vào nút mới

        head = newNode;

// Return nút đầu mới của DSLK

        return head;

}
```

5.3. CHÈN Ở CUỐI DSLK ĐƠN

Để chèn một node ở cuối danh sách, duyệt danh sách cho đến khi đến nút cuối cùng, và sau đó liên kết node mới với node cuối cùng hiện tại.



Hình 1.3: *Chèn ở cuối DSLK đơn*

Các bước tiếp cận:

- Tạo một node mới với giá trị đã cho.
- Kiểm tra nếu danh sách trống

+ Nếu đúng như vậy, tạo một node mới làm đầu và return.

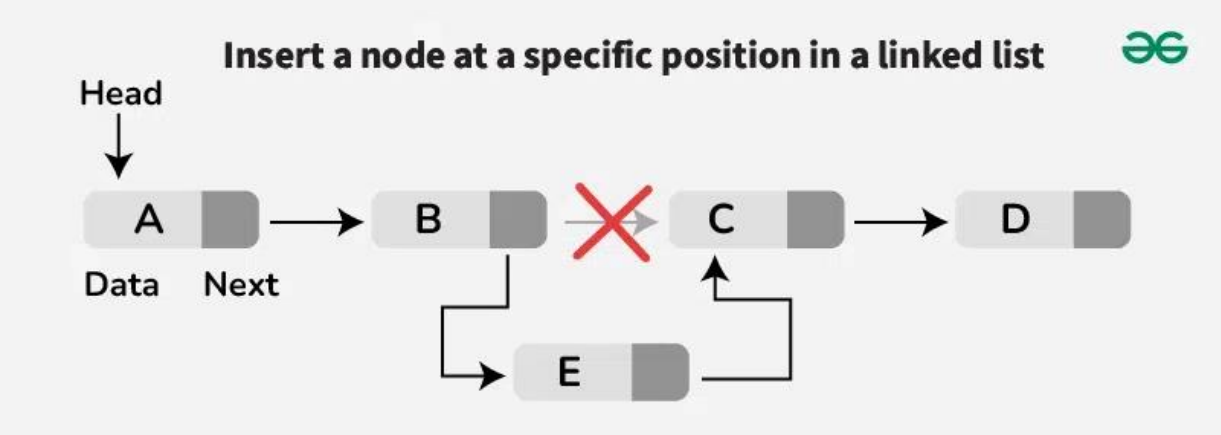
- Duyệt danh sách tới node cuối cùng
- Liên kết node mới với node cuối cùng hiện tại bằng cách đặt con trỏ next của nút cuối cùng tới nút mới

5.4. HÀM CHÈN Ở CUỐI DSLK ĐƠN

```
// Hàm C++ chèn một node mới vào cuối DSLK

Node* insertAtEnd (Node* head, int value)

{

// Tạo một node mới với giá trị cho trước

    Node* newNode = new Node(value);

// Nếu danh sách trống, tạo một node mới làm đầu

    if (head == nullptr)

        return newNode;

// Duyệt danh sách đến node cuối

    Node* curr = head;

    while (curr->next != nullptr) {

        curr = curr->next;

    }

// Liên kết node mới với node cuối hiện tại

    curr->next = newNode;

    return head; }
```

5.5. CHÈN VÀO MỘT VỊ TRÍ CỤ THỂ CỦA DSLK ĐƠN

Đề chèn một node vào một vị trí cụ thể, duyệt danh sách đến vị trí mong muốn, liên kết node mới với next node, và cập nhật liên kết tương ứng.



Hình 1.4: *Chèn vào một vị trí cụ thể của DSLK đơn*

Chúng ta chủ yếu tìm node sau đó chúng ta cần chèn node mới. Nếu chúng ta bắt gặp một giá trị NULL trước khi đến node đó, điều đó có nghĩa là vị trí cung cấp không hợp lệ.

5.6. HÀM CHÈN TẠI VỊ TRÍ CỤ THỂ CỦA DSLK ĐƠN

```
// Hàm chèn một node vào vị trí chỉ định

// Không sử dụng con trỏ kép

Node* insertPos(Node* head, int pos, int data)

{

    if (pos < 1) {

        cout << "Invalid position!" << endl;

        return head;

    }

// Trường hợp đặc biệt chèn tại đầu

    if (pos == 1) {

        Node* temp = new Node(data);

        temp->next = head;

        return temp;

    }

// Duyệt danh sách tìm node

// trước khi chèn điểm
```

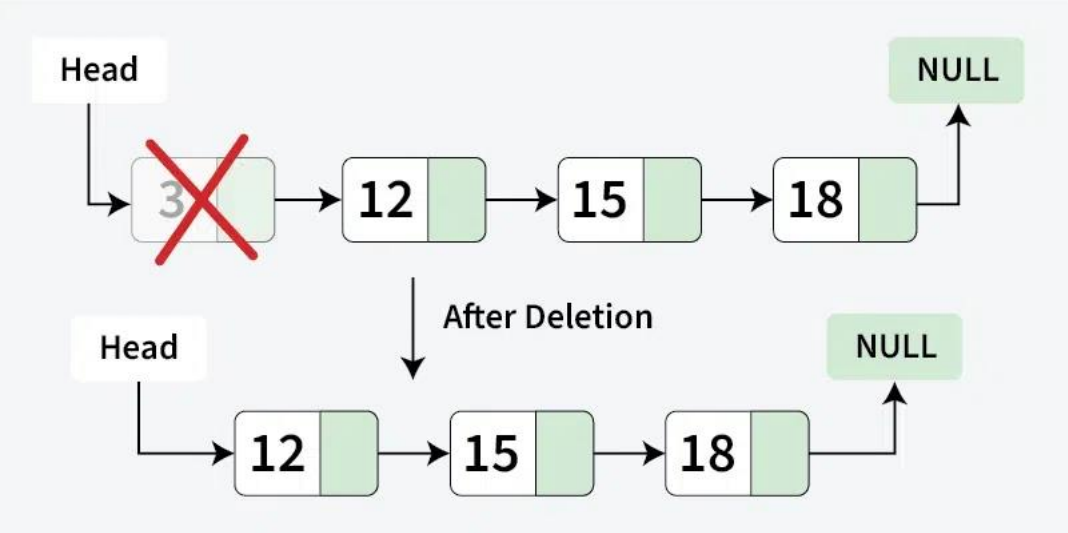
```
    Node* prev = head;
    int count = 1;
        while (count < pos - 1 && prev != nullptr) {
        prev = prev->next;
        count++;
    }
// Nếu vị trí lớn hơn số lượng node
    if (prev == nullptr) {
        cout << "Invalid position!" << endl;
        return head;
    }
// Chèn node mới tại vị trí chỉ định
    Node* temp = new Node(data);
    temp->next = prev->next;
    prev->next = temp;
    return head;
}
```

CHƯƠNG VI: XÓA TRONG DSLK ĐƠN

Xóa bao gồm việc loại bỏ một node từ DSLK. Tương tự với chèn, xóa cũng nhiều trường hợp xóa khác nhau:

6.1. XÓA Ở ĐẦU DSLK ĐƠN

Để xóa node đầu, cập nhật phần đầu để trỏ đến node thứ hai trong danh sách



Hình 1.5: *Xóa ở vị trí đầu trong danh sách*

Cách bước tiếp cận:

- Kiểm tra nếu đầu là NULL.

  + nếu nó đúng, return NULL (danh sách trống)

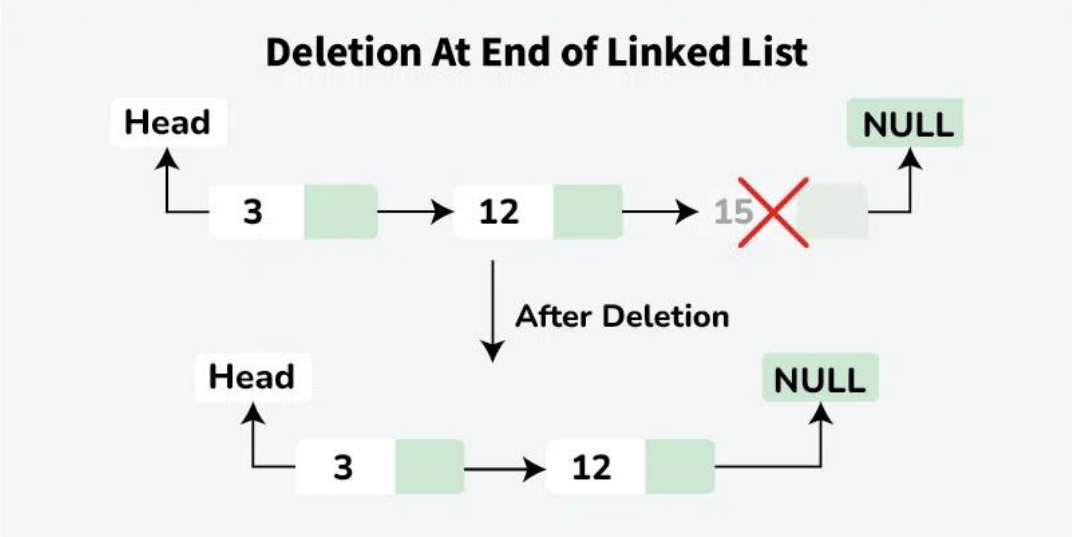
- Lưu trữ node đầu hiện tại trong một biếm tạm **temp.**
- Di chuyển đầu con trỏ với next node.
- Xóa node tạm.
- Return đầu mới của DSLK

6.2. HÀM XÓA Ở ĐẦU DSLK ĐƠN

```
// Hàm C++ xóa node đầu tiên của DSLK

Node* removeFirstNode (Node* head)

{

    if (head == nullptr)

    return nullptr;

// Di chuyển con trỏ đầu đến next node

    Node* temp = head;

    Head = head->next;

    delete temp;

    return head;

}
```

6.3. XÓA CUỐI DSLK ĐƠN

Để xóa node cuối, duyệt danh từ node thứ hai đến node cuối cùng và cập nhật trường tiếp theo của nó thành không.



Hình 1.6: *Xóa Cuối DSLK đơn*

Các bước tiếp cận:

- Kiểm tra nếu head là **NULL**.

  + Nếu đúng như vậy, return **NULL**(danh sách trống).

- Kiểm tra nếu head của **next** là **NULL** (chỉ có duy nhất 1 node trong danh sách).

  + Nếu đúng như vậy, xóa head và return **NULL.**

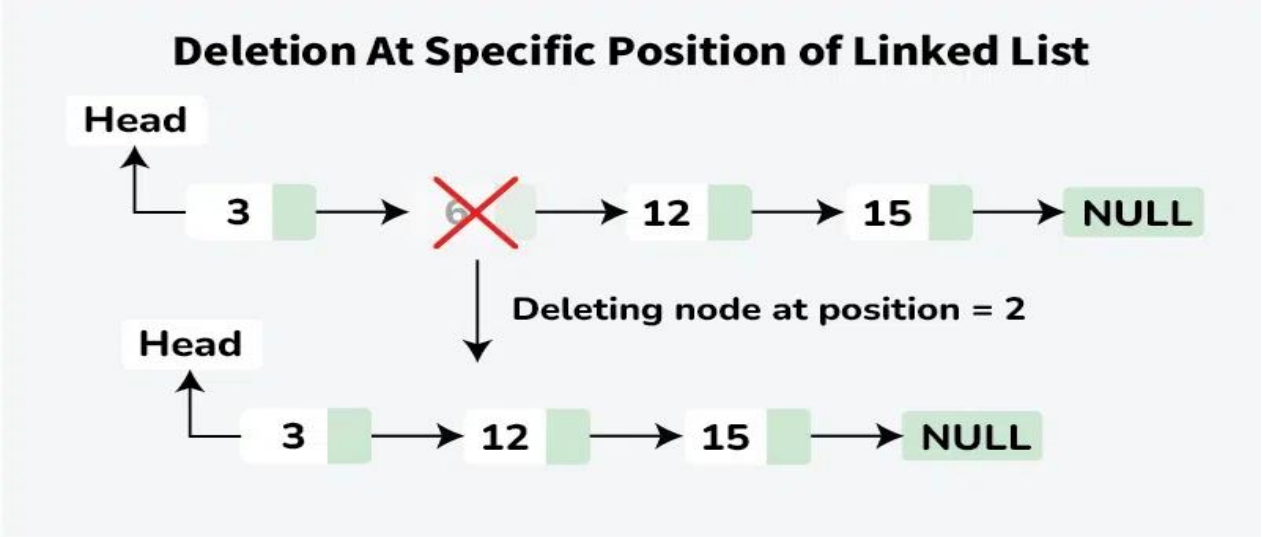
- Duyệt danh sách tìm node cuối cùng thứ hai **(second_last)**.
- Xóa node cuối **(**node sau **second_last).**
- Đặt con trỏ **next** của node cuối cùng thứ hai bằng **NULL**
- Return head của DSLK

6.4. HÀM XÓA CUỐI DSLK ĐƠN

```
// Hàm C++ xóa node cuối của DSLK

Node* removeLastNode (Node* head)

        if (head == nullptr) return nullptr;

        if (head->next == nullptr) {

        delete head;

        return nullptr;

        }

// Tìm nút cuối thứ hai

        Node* second_last = head;

        while (second_last->next->next != nullptr)

                second_last = second_last->next;

// Xóa node cuối

        delete (second_last->next);

// Thay đổi next của second last

        second_last->next = nullptr;|

        return head; }
```

6.5. XÓA TẠI VỊ TRÍ CỤ THỂ CỦA DSLK ĐƠN

- Để xóa một node tại vị trí cụ thể, duyệt danh sách vị trí mong muốn, cập nhật liên kết truyền cập nhật các liên kết để bỏ qua node bị xóa.



Hình 1.7: *Xóa một DSLK node tại vị trí chỉ định*

Các bước tiếp cận:

- Kiểm tra nếu danh sách trống hoặc vị trí không hợp lệ, return nếu có.
- Nếu head cần xóa, cập nhật head và xóa node.
- Duyệt node trước vị trí cần xóa.
- Nếu vị trí vượt ngoài phạm vi, return.
- Lưu trữ node sẽ bị xóa
- Cập nhật các liên kết để bỏ qua node.
- Xóa node được lưu trữ.

6.6. HÀM XÓA TẠI VỊ TRÍ CỤ THỂ CỦA DSLK ĐƠN

```
// Hàm C++ xóa một node tại một vị trí cụ thể

Node* deleteAtPosition(Node* head, int position)

{

// Nếu danh sách trống hoặc vị trí không hợp lệ

    if (head == nullptr || position < 1) {

        return head;

    }

// Nếu head cần xóa

    if (position == 1) {

        Node* temp = head;

        head = head->next;

        delete temp;

        return head;

    }

// Duyệt node trước vị trí cần xóa

    Node* current = head;
```

```
        for (int i = 1; i < position - 1 && current != nullptr; i++) {
            current = current->next;
        }
// Nếu vị trí vượt ngoài phạm vi
        if (current == NULL || current->next == nullptr) {
            return;
        }
// Lưu trữ node sẽ bị xóa
        Node* temp = current->next;
// Cập nhật các liên kết để bỏ qua node cần xóa
        current->next = current->next->next;
// Xóa node
        delete temp;
        return head;
}
```

CHƯƠNG VII: KẾT LUẬN

Việc cài đặt danh sách liên kết đơn trên mảng không chỉ mang lại một cách tiếp cận sáng tạo mà còn giúp hiểu rõ hơn về bản chất của cấu trúc dữ liệu. Phương pháp này cho thấy sự linh hoạt khi sử dụng bộ nhớ tĩnh để mô phỏng các liên kết, đồng thời giúp giải quyết nhiều bài toán trong lập trình mà không cần đến con trỏ hoặc bộ nhớ động.

Qua việc nghiên cứu và thực hiện chủ đề này, nhóm chúng em đã có cơ hội áp dụng lý thuyết vào thực tế, phát triển tư duy lập trình và tìm hiểu sâu hơn về cách quản lý dữ liệu hiệu quả. Chúng tôi tin rằng, không chỉ đối với nhóm mà cả những ai quan tâm đến cấu trúc dữ liệu đều sẽ nhận được những bài học giá trị từ chủ đề này.

Danh sách liên kết trên mảng là minh chứng cho sự sáng tạo trong khoa học máy tính và sẽ luôn là một giải pháp đáng cân nhắc trong việc tối ưu hóa và tổ chức dữ liệu.

DANH MỤC TÀI LIỆU THAM KHẢO

[1]. Trần Quý Nam *(2025), Tài liệu học tập , CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT, Khoa CNTT - Trường Đại học Đại Nam.*

[2] Trần Quý Nam *(2025), Bài giảng CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT, Khoa CNTT - Trường Đại học Đại Nam.*